Title : Multimodal Emotion Recognition using Speech and Text

Problem statement: To design a system that recognizes emotions by learning and combining emotional information from speech and text .

Introduction

Emotion recognition is an important task in computing , enabling machines to understand human emotions for applications like intelligent assistants.

Dataset Description

Dataset: Toronto Emotional Speech Set (TESS)

- Source (Kaggle -TESS)
- No of samples: 2800 samples
- Emotion classes : sad , pleasant surprised , neutral , happy , fear , disgust, angry
- English speech recordings spoken by two female actors:
  - OAF : Older Adult Female
  - YAF: Younger Adult Female
- Each emotion class has 400 samples.
- The dataset contains speech recordings saved as .wav files
  - Emotion labels are encoded in the file names.
- The dataset does not contain any additional transcript
  - The audio file will be about one word with an additional sentence with emotion class.
  - This additional sentence is same for all the audio files
  - The spoken word will be the audio file name followed by emotion class
    - For example: YAF_thumb_sad.wav
    - Here thumb is the spoken word .
    - Sad is the emotional class.
    - YAF is the speaker.
- So we need to explicitly extract the transcripts.
- Dataset split :
  - Train : 70%
  - Validation: 15%
  - Test : 15%

Data preprocessing

- Dataset Loading:
  - If you are using colab we can load the kaggle dataset directly from data explorer in colab notebook instead of downloading the whole dataset.
  - If not we need to download the dataset from kaggle and dataset should be inside the working folder.
- Extraction of audio features:
  - To extract emotion cues in the given audio , I used pre- trained HuBERT - Base model.

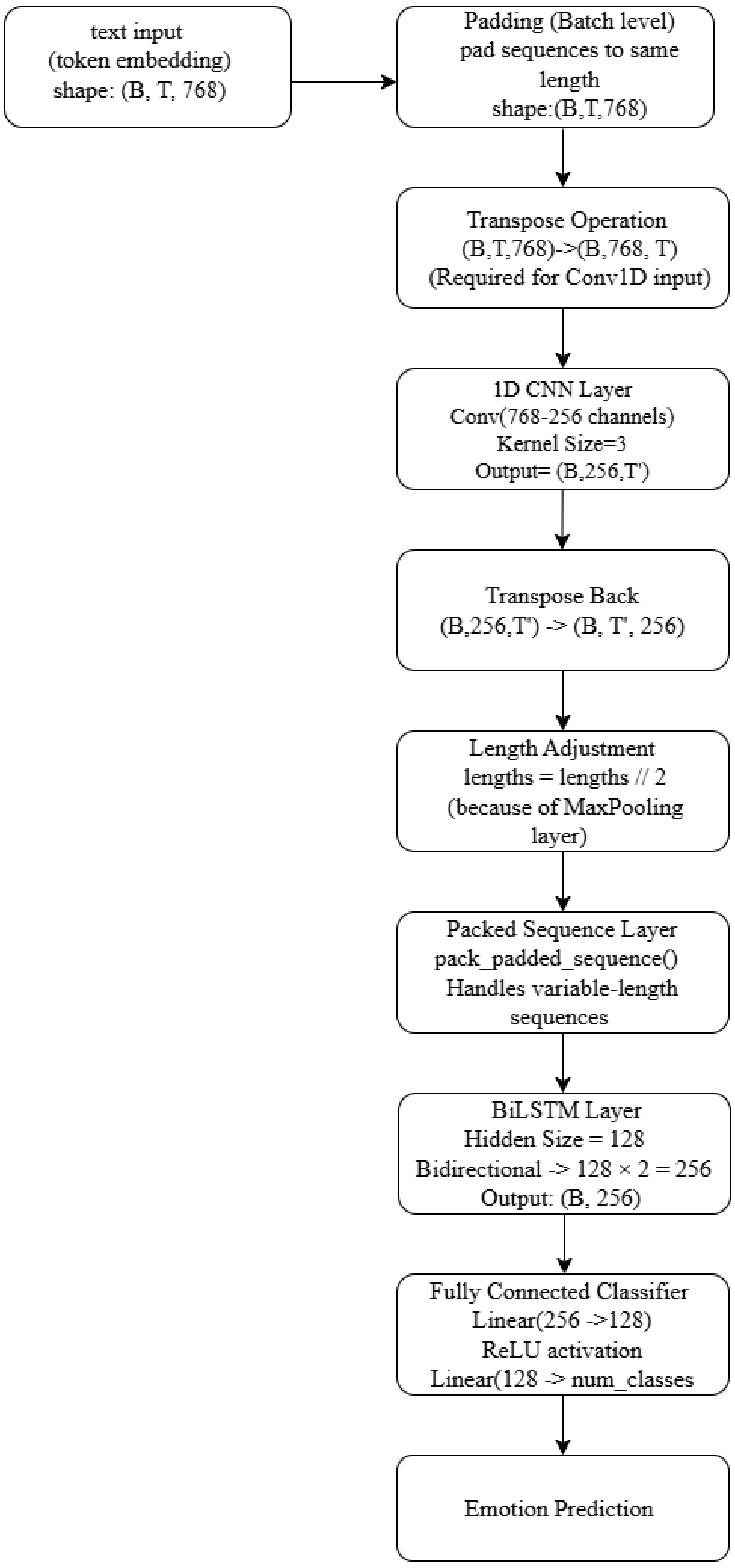
        - In HuBERT , I preferred using only a specific transformer layers (layer 5 to layer 9) to extract emotion cues .
        - The pretrained HuBERT model is frozen and used as the feature extractor to prevent the overfitting on small datasets.
    - Here the audio given is resampled to 16kHz (required sampling rate for HuBERT).
    - After resampling the audio is converted into appropriate input format using HuggingFace’s Wav2Vec2FeatureExtractor before passing into HuBERT model
    - Then audio is passed to HuBERT model for feature extraction and saved extracted features in the format of .pt files inside the output folder (named hubert_features).
    - The output folder has 7 folders inside it , in which each folder represents each emotion class
    - This whole process is done by using python script (audio_extraction.ipynb)

- Extraction of text features
    - To extract linguistic emotion cues in the given speech dataset , I used pre-trained BERT - Base model (bert-base-uncased) from HuggingFace.
    - In BERT , I preferred using only a specific transformer layers (6 layers) to extract emotion cues
    - The pre-trained BERT is frozen and used only as a feature extractor to prevent overfitting.
    - In the TESS dataset, each audio file contains a single spoken word. The spoken word is extracted directly from the filename using a custom function.
    - HuggingFace BertTokenizer is used for tokenization of words and addition of special tokens ([CLS] word[SEP]).
    - These tokens are passed to BERT for feature extraction and extracted features are saved in the format of .pt files inside the output folder (named text_ features)
    - This whole process is done by using python script (text_extraction.py)
- Now folders hubert_features(for audio) and text_features(for text) will be the inputs for model building.

ARCHITECTURE

Text-only model

- Input given in this model is only text_features and here we do not involve speech for classification.
- The input is given to CNN layer(conv1D) and then given to BiLSTM layers (temporal modelling).
- The output from these layers are given to the Classifier layer for prediction.
- This whole process is done using python script (text_only_model).

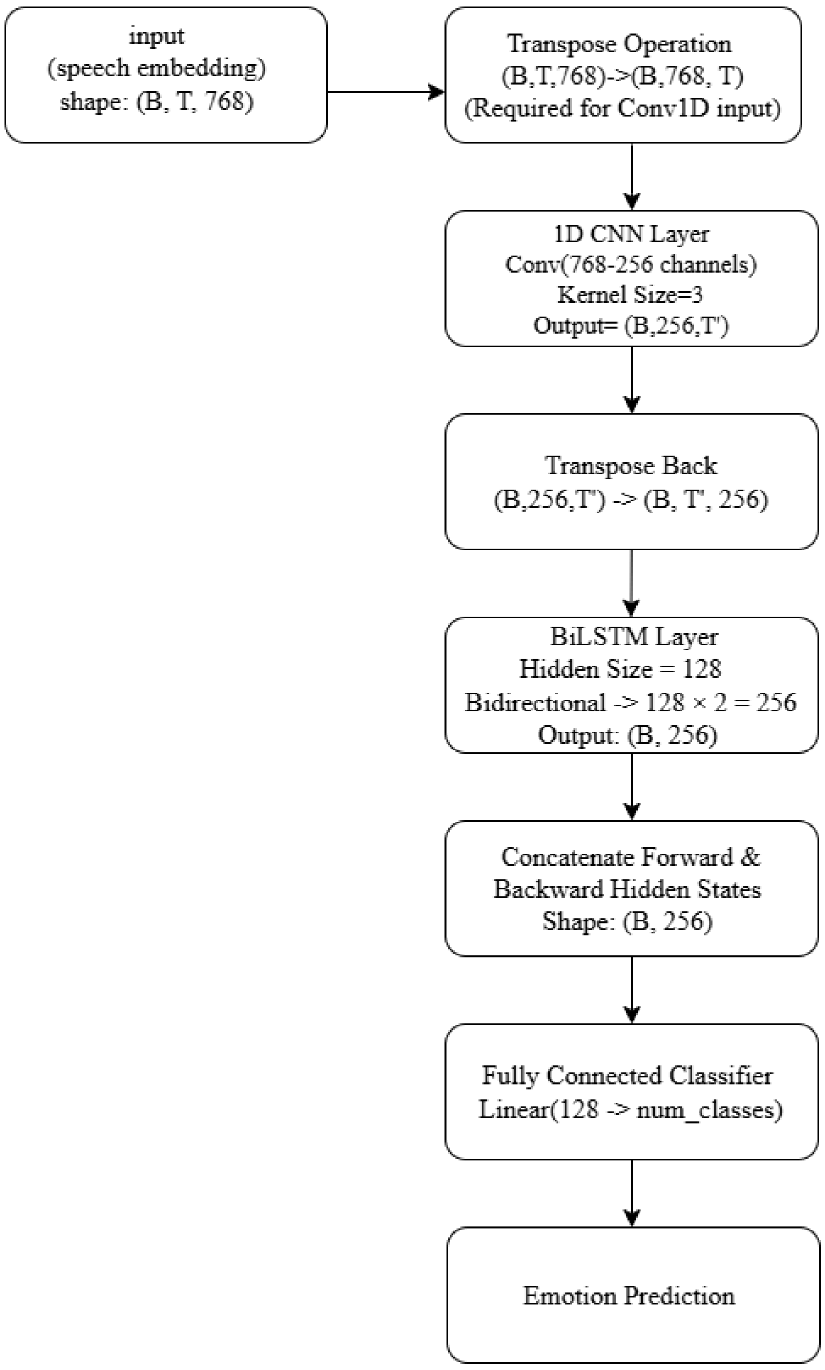

text input
(token embedding)
shape: (B, T, 768)
Padding (Batch level)
pad sequences to same length
shape:(B,T,768)
Transpose Operation
(B,T,768)->(B,768, T)
(Required for Conv1D input)
1D CNN Layer
Conv(768-256 channels)
Kernel Size=3
Output= (B,256,T')
Transpose Back
(B,256,T') -> (B, T', 256)
Length Adjustment
lengths = lengths // 2
(because of MaxPooling layer)
Packed Sequence Layer
pack_padded_sequence()
Handles variable-length sequences
BiLSTM Layer
Hidden Size = 128
Bidirectional -> 128 × 2 = 256
Output: (B, 256)
Fully Connected Classifier
Linear(256 ->128)
ReLU activation
Linear(128 -> num_classes
Emotion Prediction

Why this Architecture:

- BERT:
    - It is used because it provides contextual word representations learned from large scale language corpus.
    - Unlike other methods BERT produces high dimensional embeddings to capture relationship between the tokens effectively
- CNN(conv 1D):
    - It captures local patterns in the input sequence between neighbouring tokens which are important for emotion cues.
    - CNN can take input as high dimensional input features and transform them into more compact representations.
    - It provides refined feature representations that improves the effectiveness of the BiLSTM layer that follows.
- Max pooling
    - Reduces sequence length to speed up training.
    - To enhance temporal modeling efficiency.
    - To prevent overfitting.
- Packed Sequences:
    - Handles variable length inputs effectively
    - Ensures that padded zeros do not affect hidden state updates .
- BiLSTM:
    - It captures sequential relationships between the data.
    - Unlike lstm, Bilstm processes information in both forward and backward directions, allowing the model to use past and future context simultaneously.
    - Handles variable length sequences i.e Bilstm naturally supports sequences of different lengths making them suitable for text and speech inputs.
    - Uses memory gates to retain important information over long sequences , where conventional RNN's struggle.
    - After CNN extracts local patterns ,BiLSTM learns temporal relationships across the sequence .
- Classifier:
    - Converts learned features into probability scores for each emotion class.

Speech - Only Model:

- Input given to this model is the hubert_features(audio) folder and here we do not involve transcripts or text features here.
- The input is given to CNN layer(conv1D) and then given to BiLSTM layers (contextual modelling).
- The output from these layers are given to the Classifier layer for prediction.
- This whole process is done using a python script (speech_only_model.ipynb).



Why this Architecture:

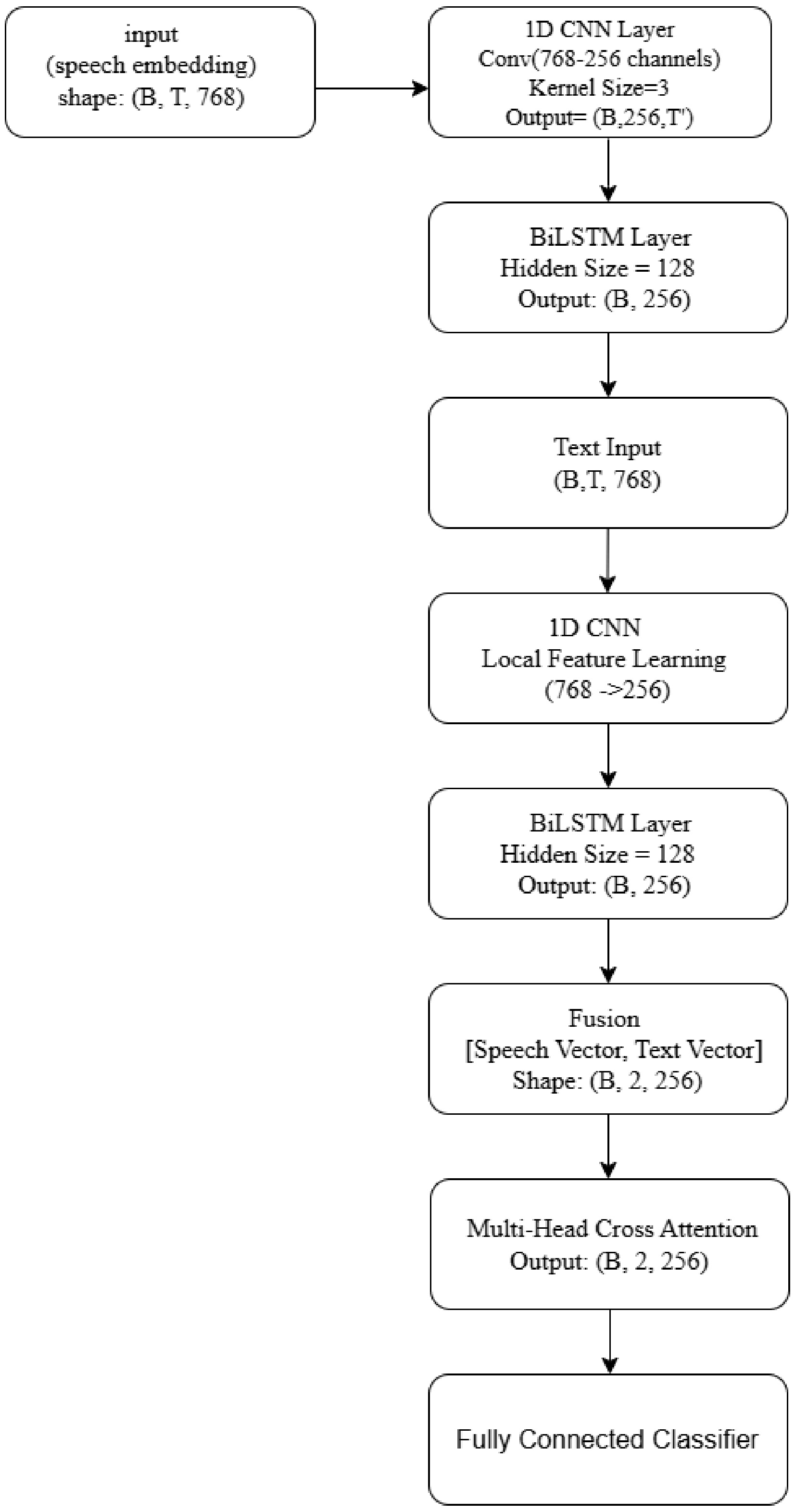
- HuBERT:
    - It is pretrained on large-scale speech datasets and learns rich acoustic representations.
    - Captures high level information like pitch tone , emotion cues.
    - Suitable for small datasets that prevents overfitting.
- CNN:
    - It captures local patterns in the input sequence between neighbouring frames which are important for emotion cues.

    - CNN can take input as high dimensional input features and transform them into more compact representations.
    - It provides refined feature representations that improves the effectiveness of the BiLSTM layer that follows.
- BiLSTM:
    - It captures sequential relationships between the data.
    - Unlike lstm, Bilstm processes information in both forward and backward directions, allowing the model to use past and future context simultaneously.
    - Handles variable length sequences i.e Bilstm naturally supports sequences of different lengths making them suitable for text and speech inputs.
    - Uses memory gates to retain important information over long sequences , where conventional RNN's struggle.
    - After CNN extracts local patterns ,BiLSTM learns temporal relationships across the sequence .
- Classifier:
    - Converts learned features into probability scores for each emotion class.

Speech+text model :

- Input given here is both text and speech features .
- Here we combine 2 models one is text_only_model and Speech_only_model.
- By combining these two models ,the output generated in these two models are then done fusion using cross attention.
- This makes the model make the prediction by both audio and transcript basis.
- This whole process is done using python script (fusion.py)

-



-

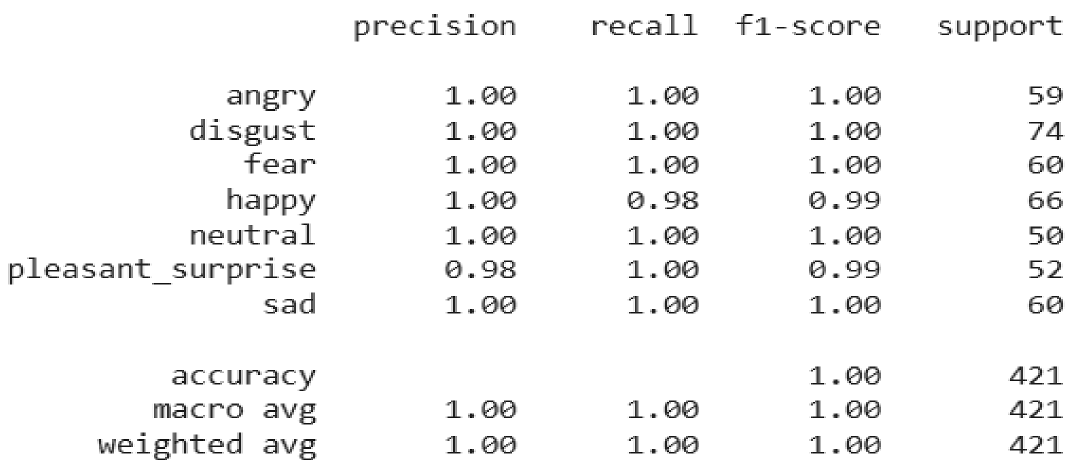
Why this architecture:

- BERT:
    - It is used because it provides contextual word representations learned from large scale language corpus.
    - Unlike other methods BERT produces high dimensional embeddings to capture relationship between the tokens effectively.

- HuBERT:
    - It is pretrained on large-scale speech datasets and learns rich acoustic representations.
    - Captures high level information like pitch tone , emotion cues.
    - Suitable for small datasets that prevents overfitting.
-
- CNN(conv 1D):
    - It captures local patterns in the input sequence between neighbouring tokens or frames which are important for emotion cues.
    - CNN can take input as high dimensional input features and transform them into more compact representations.
    - It provides refined feature representations that improves the effectiveness of the BiLSTM layer that follows.
- BiLSTM:
    - It captures sequential relationships between the data.
    - Unlike lstm, Bilstm processes information in both forward and backward directions, allowing the model to use past and future context simultaneously.
    - Handles variable length sequences i.e Bilstm naturally supports sequences of different lengths making them suitable for text and speech inputs.
    - Uses memory gates to retain important information over long sequences , where conventional RNN’s struggle.
    - After CNN extracts local patterns ,BiLSTM learns temporal relationships across the sequence .
- Cross Attention:
    - Learns relationships between modalities (speech and text)
    - The model automatically learns which modality is more important for each sample instead of assigning equal importance.
    - It produces richer and more informative fused features compared to simple concatenation or averaging.
- Classifier:
    - Converts learned features into probability scores for each emotion class.

Comparison on three models :

- **Speech only model**

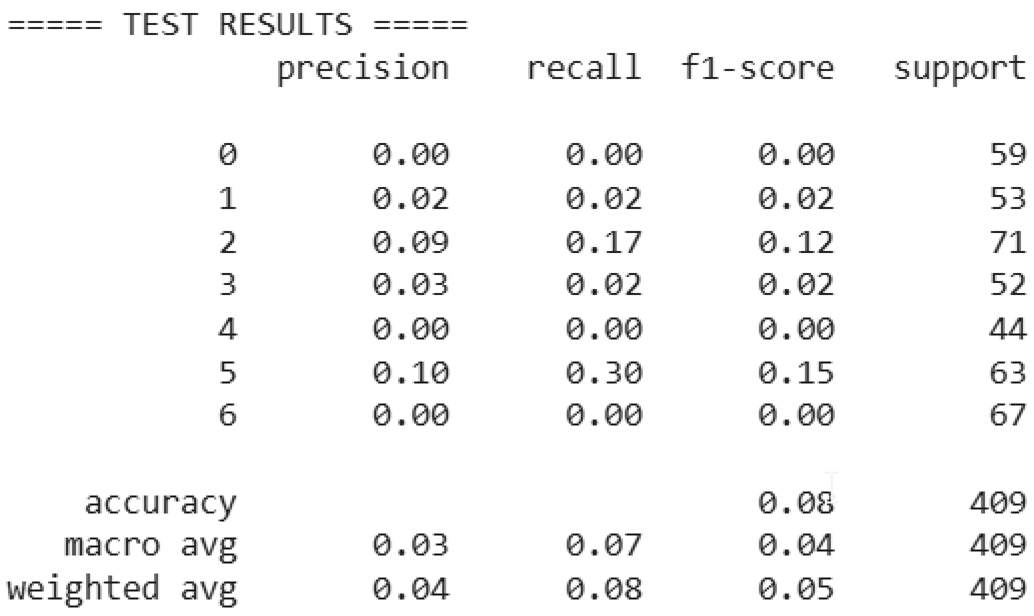


```
                  precision    recall  f1-score   support

           angry       1.00      1.00      1.00        59
         disgust       1.00      1.00      1.00        74
            fear       1.00      1.00      1.00        60
           happy       1.00      0.98      0.99        66
         neutral       1.00      1.00      1.00        50
pleasant_surprise       0.98      1.00      0.99        52
             sad       1.00      1.00      1.00        60

        accuracy                           1.00       421
       macro avg       1.00      1.00      1.00       421
    weighted avg       1.00      1.00      1.00       421
```

- **Text_only model**

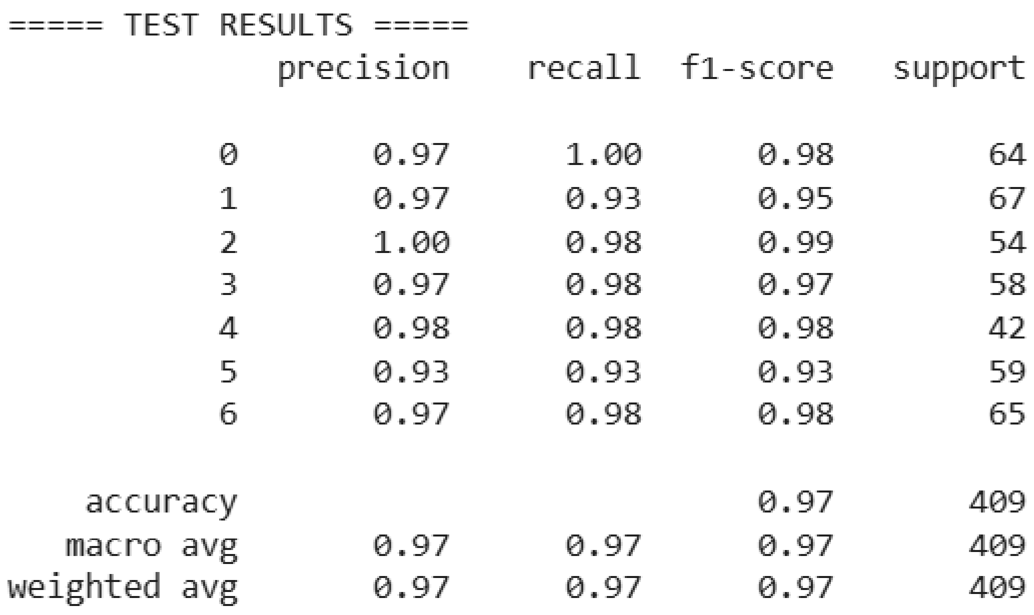


```
===== TEST RESULTS =====
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        59
           1       0.02      0.02      0.02        53
           2       0.09      0.17      0.12        71
           3       0.03      0.02      0.02        52
           4       0.00      0.00      0.00        44
           5       0.10      0.30      0.15        63
           6       0.00      0.00      0.00        67

    accuracy                           0.08       409
   macro avg       0.03      0.07      0.04       409
weighted avg       0.04      0.08      0.05       409
```

- **Multimodal fusion**



```
===== TEST RESULTS =====
              precision    recall  f1-score   support

           0       0.97      1.00      0.98        64
           1       0.97      0.93      0.95        67
           2       1.00      0.98      0.99        54
           3       0.97      0.98      0.97        58
           4       0.98      0.98      0.98        42
           5       0.93      0.93      0.93        59
           6       0.97      0.98      0.98        65

    accuracy                           0.97       409
   macro avg       0.97      0.97      0.97       409
weighted avg       0.97      0.97      0.97       409
```

Observation:

- The **text-only model** demonstrated comparatively lower performance since the TESS dataset contains mostly single-word transcripts with limited emotional information.
- The **speech-only model** achieved great performance because emotional cues are primarily conveyed through vocal characteristics such as tone, pitch.

- The **multimodal fusion model** achieved the highest accuracy, indicating that combining speech and text features provides information that enhances emotion recognition.

Analysis:
Which emotions are easiest / hardest to classify? Why?

- Easy: From the **speech-only** and **multimodal fusion** results:
    - Angry
    - Disgust
    - Fear
    - Sad
    - Neutral
- Reasons:
    - Strong acoustic cues:
        - Angry: high energy, high pitch
        - Sad: low pitch
        - Fear : irregular energy
    - Emotional intensity differences make temporal modeling easier.
    - These emotions are less dependent on text meaning.
- Hardest Emotions:
    - Happy'
    - Pleasant surprise
- Reasons :
    - Happy and surprise both have:
        - High pitch
        - High energy
        - Fast speaking rate
    - Surprise often depends on semantic content not just on tone.

- Text-only Model Failure
    - Model accuracy = 12%
    - This is extremely low because :
        - TESS dataset sentences are emotion- neutral phrases i.e
            - Example: day the word dog
        - No emotional words so bert cannot learn emotion

When does fusion help most?

- Fusion helps when :
    - Speech signal alone is hard to predict i.e
        - Happy vs Surprise
        - Neutral vs Pleasant
        - Because speech gives about emotional intensity and text tells about semantic cues.
        - Fusion combines emotion+context= better prediction.

Error Analysis (3–5 Failure Cases)

- **Text only model**
    - True emotion is fear and predicted emotion is happy
    - True emotion is sad and predicted emotion is happy
    - True emotion is angry and predicted emotion is happy

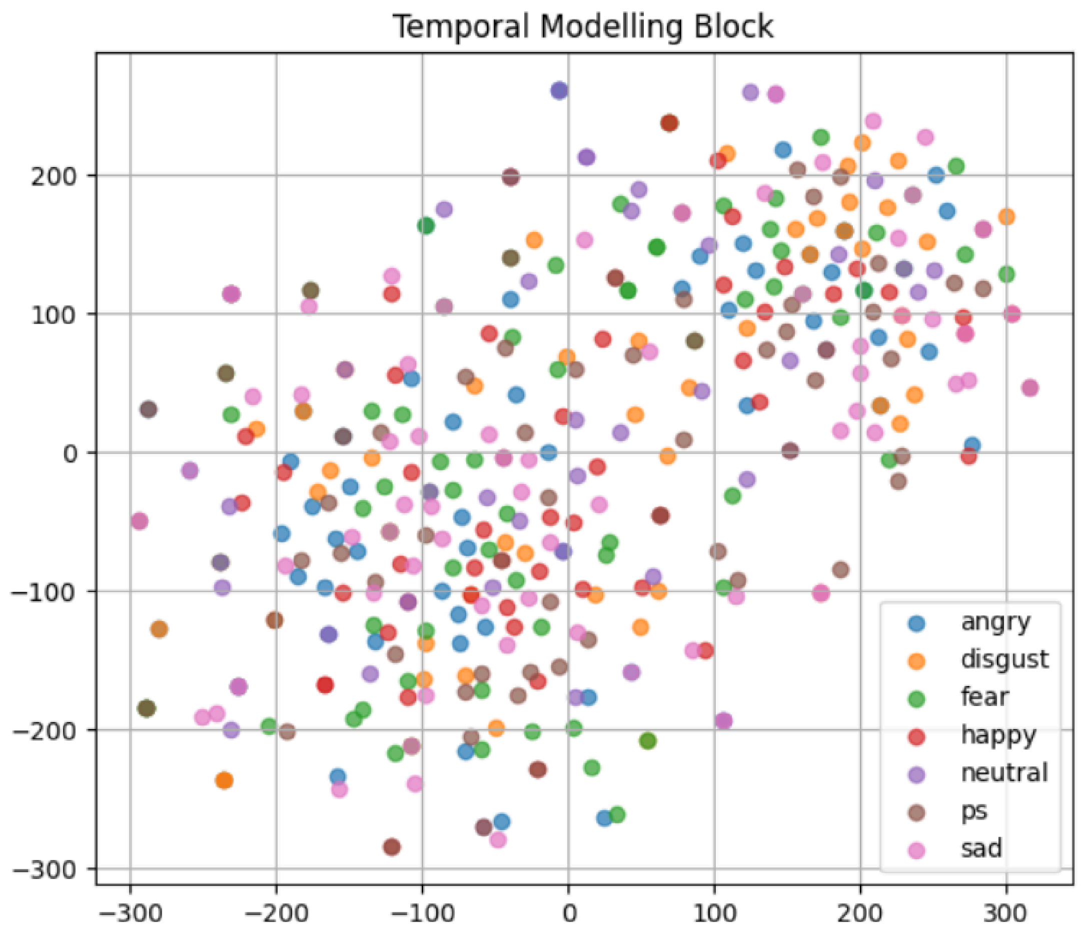
Semantic information does not have any information about emotion because transcripts are very short telling about only one word which does not refer the emotion hence we see the prediction is wrong.

- **Fusion model**
    - True emotion is ps and predicted emotion is disgust
    - True emotion is surprise and predicted emotion is neutral

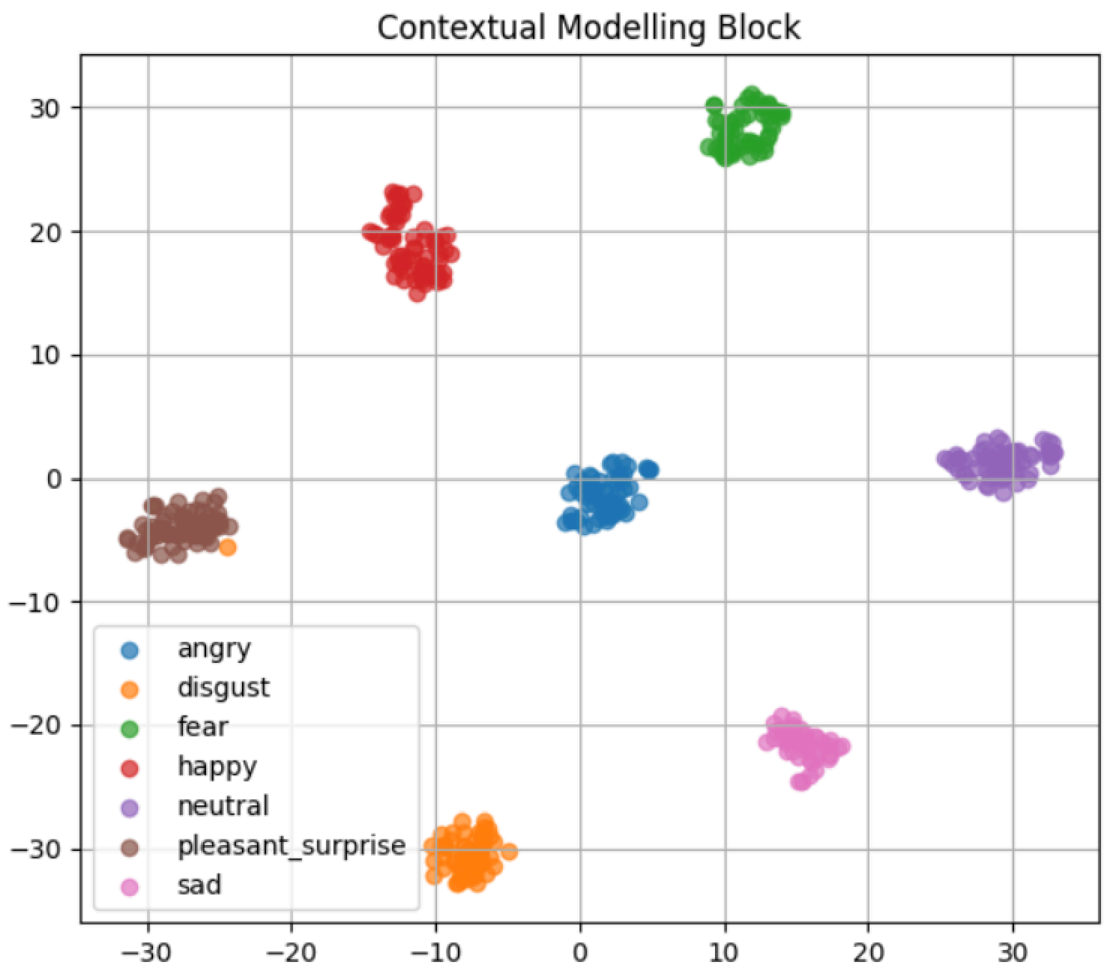
This is because of the semantic influence of the text over speech during fusion. Maybe it depends on whether the word may be referring to a particular scenario and audio is expressing the emotion which may not be similar.

Visualize the separability of emotion clusters using the learned representations from:

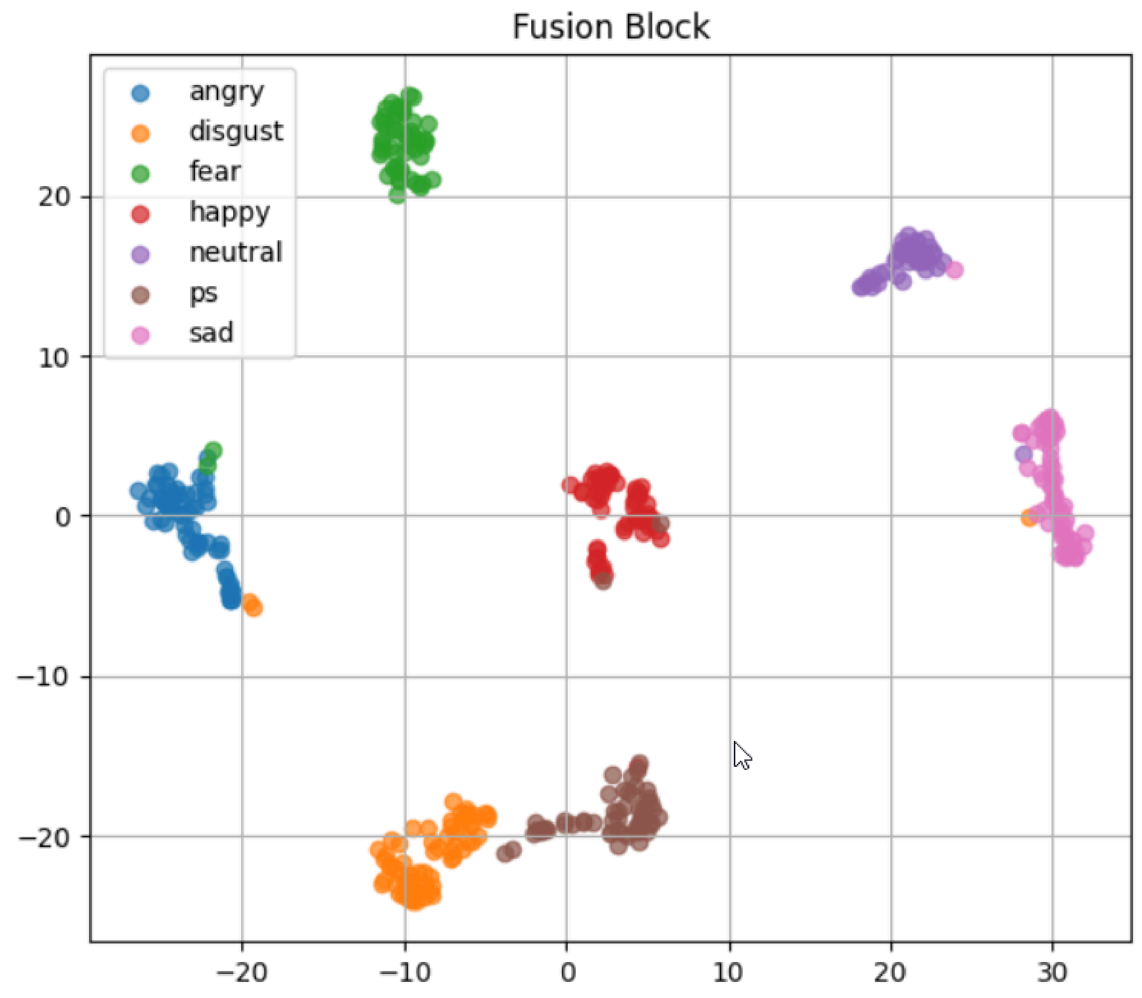
- Temporal modelling block



- **Contextual modelling**


Contextual Modelling Block
angry
disgust
fear
happy
neutral
pleasant_surprise
sad
30
20
10
0
−10
−20
−30
−30
−20
−10
0
10
20
30

-
- **Fusion**



Fusion Block
angry
disgust
fear
happy
neutral
ps
sad
20
10
0
−10
−20
−20
−10
0
10
20
30

•